robotics

MDPI

Article

Nonlinear Model Predictive Control for Mobile Robot Using Varying-Parameter Convergent Differential Neural Network

Yingbai Hu [1], Hang Su [2], Longbin Zhang [3], Shu Miao [4], Guang Chen [1,5,*] and Alois Knoll [1]

1 Department of Informatics, Technical University of Munich, 85748 Munich, Germany
2 Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milano, Italy
3 BioMEx Center & KTH Mechanics, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden
4 School of Mechanical Engineering and Automation, Harbin Institute of Technology, Shenzhen 518052, China
5 College of Automotive Engineering, Tongji University, Shanghai 201804, China
* Correspondence: guangchen@tongji.edu.cn; Tel.: +86-13671534131

Received: 29 June 2019; Accepted: 26 July 2019; Published: 31 July 2019

check for updates

Abstract: The mobile robot kinematic model is a nonlinear affine system, which is constrained by velocity and acceleration limits. Therefore, the traditional control methods may not solve the tracking problem because of the physical constraint. In this paper, we present the nonlinear model predictive control (NMPC) algorithm to track the desired trajectory based on neural-dynamic optimization. In the proposed algorithm, the NMPC scheme utilizes a new neural network named the varying-parameter convergent differential neural network (VPCDNN) which is a Hopfifield-neural network structure with respect to the differential equation theory to solve the quadratic programming (QP) problem. The new network structure converges to the global optimal solution and it is more efficient than traditional numerical methods. In the simulation, we verify that the proposed method is able to successfully track reference trajectories with a two-wheel mobile robot. The experimental validation has been conducted in simulation and the results show that the proposed method is able to precisely track the trajectory maintaining a high robustness based on the VPCDNN solver.

Keywords: mobile robot; nonlinear model predictive control; quadratic programming; varying-parameter convergent differential neural network

1. Introduction

Over past decades, the motion control of mobile robots has increasingly attracted a lot of researchers, especially for applications in industry and transportation. The two-wheel mobile robot is a special differential wheeled robot and the physical system is the nonholonomic system, so the two-wheel robot system is in asymptotic stabilization when it is controlled by time-varying methods based on Brockett's theory [1].

There are various approaches that have been developed for mobile robot control and in some applications under certain conditions. In Reference [2], the authors presented the back-stepping control algorithm for the curved weld seam tracking of a mobile robot and achieved good tracking precision. In Reference [3], a nonlinear control algorithm based on sliding mode control (SMC) was developed for mobile robots, where the four wheel motion mode was converted to an inverted pendulum mode, which has a small turning radius and high-speed observation features. In Reference [4], the iterative learning control method was developed for path-tracking, which can reduce the errors in Cartesian space for repeat motions. In Reference [5], the fuzzy control is applied to autonomous mobile robot control. In Reference [6,7], the author presents a robust adaptive feedback controller for a tractor trailer wheeled robot under uncertain parameters.

In Reference [8], a non-iterative controller is presented for a nonlinear mobile robot system without constraint. The new method has a low worst-cost ration than nonlinear model predictive control method (NMPC), but cannot ensure safety in its running because no physical limits are considered. In Reference [9], an adaptive optimal control method is presented to solve the $H\infty$ problem under changing environmenalt conditions, which can adapt to a complex environment, but it seems that the gradient-descent based solver is used to solve the optimal problem.

For safety reasons, the velocity and acceleration constraints (equivalent to velocity incremental constraint) and other constraints need to be taken into account during operation, which is not suitable for some special applications with physical limits and is very dangerous in real time, running over the limit speed. In this paper, the nonlinear model predictive control algorithm is employed to optimize the performance of the mobile robot during the tracking of the reference trajectories. Due to the excellent performance of the NMPC control method with nonlinear input-output over a finite horizon, it is widely used to track trajectories with the kinematic constraints of a mobile robot.

There are many traditional model predictive control methods developed for the tracking of mobile robots. In References [10,11], the linear model predictive control is used for tracking problems with fast small-scale solvers. However, for strong nonlinear robot systems, linear models find it difficult to describe the complex motion process, and it is hard to predict the next step accurately. Therefore, in this paper, the nonlinear model predictive control method is proposed to control the mobile robot system. Compared with the linear MPC method, NMPC describes the complex dynamic processes of the robot system more accurately.

The NMPC method has the nonlinear input-output over a finite horizon and it is widely used for tracking in mobile robots or auto driving vehicle or mobile robots [12,13]. In Reference [14], NMPC is applied to control vehicle velocity which demonstrated the NMPC's energy cost.

In References [15,16], a novel NMPC method is presented to control vehicles, which has complex nonlinear terms and uncertainties. In References [17,18], the stochastic MPC method is employed to penalize undesired factors. In References [19,20], the traditional numerical optimization method is used for solving the quadratic programming (QP) problem of MPC. However, it has the computational burden problem which cannot meet online computing in a real time running application.

In Reference [21], the authors present the NMPC method to follow customers and maintain a certain distance, but the solver for NMPC seems to be based on the gradient-descent method. In this case, the methods cannot track the theoretical solutions because the theoretical solutions change over time, which causes an undesired performance and cannot converge to 0 in a limited amount of time. Therefore, time-varying neural network methods have been designed to solve the optimization problem of NMPC.

There are many works on the neural network methods for solving MPC optimization [22]. In Reference [23], the nonlinear model predictive control scheme using general projection neural networks (GPNN) is employed to optimize the chained systems with nonholonomic constraints. The GPNN is designed for the QP problem and it globally converges to the optimal solution with a quick convergence rate and a low computational burden. In addition, in References [24–26], the primal dual neural network (PDNN) is used to optimize the QP problem of NMPC, where the PDNN has no matrix inversion and matrix-matrix multiplication, which is suitable for the online optimization in References [27,28].

However, these are all time-invariant methods based on gradient-descent for optimization. Inspired by the works mentioned above, in this paper a new neural network named the varying-parameter convergent differential neural network (VPCDNN) method is employed to solve the QP problems of MPC.

Our VPCDNN has the same neurons in the network as the decision variables for NMPC optimization. The proposed VPCDNN can obtain an optimal solution for the QP problem in real-time application.

The main contributions of this paper are:

1. A varying parameter convergent differential neural network method is proposed to solve the time varying QP problem of MPC and all state variables can converge quickly to the optimal value using the neural network with physical. This is the first time that be presented to optimize the MPC problem.
2. The convergence analysis of VPCDNN and the simulation results demonstrate good performance on the convergence speed and robustness of VPCDNN.

The remainder of this paper is organized as follows. The kinematic model of mobile robots and the nonlinear model predictive control scheme are described in Section 2. The proposed VPCDNN method for solving the online QP problem of MPC is detailed in Section 3. The convergence and robustness analysis of VPCDNN is shown in Section 4. Simulation results are reported and discussed in Section 5. Finally, Section 6 concludes this paper.

2. Model Predictive Control Scheme

In this section, the kinematic model of a mobile robot is reformulated as a nonlinear affine system, then the model predictive control scheme is presented for the tracking problem. The varying-parameter neural network base model predictive control scheme is proposed to solve the QP problem with physical constraints.

2.1. Mobile Robot Control System

The kinematic model of two wheels mobile robot are shown in Figure 1. According to the relationship between robot velocity v and two driving wheels velocity (v_L, v_R), the robot velocity and angle velocity are expressed as: $v = (v_L + v_R)/2$, $\omega = (v_L - v_R)/l_f$, respectively. l_f denotes the distance of two wheels. The kinematic model formulation of mobile robot is expressed as,

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} u \tag{1}$$

where (x, y) is the mobile robot position in Cartesian space and θ is the orientation; $u = (v, \omega)^T$ is the control input and X is the state vector. From the formulation in (1), we can obtain a kinematic model of the reference trajectory by state vector $X_r = (x_r, y_r, \theta_r)^T$ and control input $u_r = (v_r, \omega_r)^T$ and it is expressed as:

$$\dot{X}_r = \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} v_r\cos\theta_r \\ v_r\sin\theta_r \\ \omega_r \end{bmatrix} = \begin{bmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{bmatrix} u_r \tag{2}$$

Therefore, we can get the robotic kinematic errors,

$$X_e = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix} \tag{3}$$

where $X_e = [x_e, y_e, \theta_e]^T$. The derivative of $\dot{X}_e$ can be obtained from the error state in (3),

$$\begin{cases} \dot{x}_e = \omega y_e - v + v_r \cos\theta_e \\ \dot{y}_e = -\omega x_e + v_r \sin\theta_e \\ \dot{\theta}_e = \omega_r - \omega \end{cases} \tag{4}$$

The control inputs are reformulated as:

$$u_e = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} v_r \cos\theta_e - v \\ \omega_r - \omega \end{bmatrix} \tag{5}$$

Substituting (5) into (4), the kinematic model of X_e can be rewritten as

$$\dot{X}_e = \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} 0 & \omega & 0 \\ -\omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} + \begin{bmatrix} u_1 \\ v_r \sin\theta_e \\ u_2 \end{bmatrix} \tag{6}$$

The formulation in (6) can be reformulated as the following equation:

$$\dot{\bar{X}}_e = \begin{bmatrix} 0 & \omega_r & 0 \\ -\omega_r & 0 & v_r \\ 0 & 0 & 0 \end{bmatrix} \bar{X}_e + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} u_e \tag{7}$$

Therefore, the motion control problem of the mobile robot in (1) is converted into a stabilization problem of a nonlinear affine system in (7).

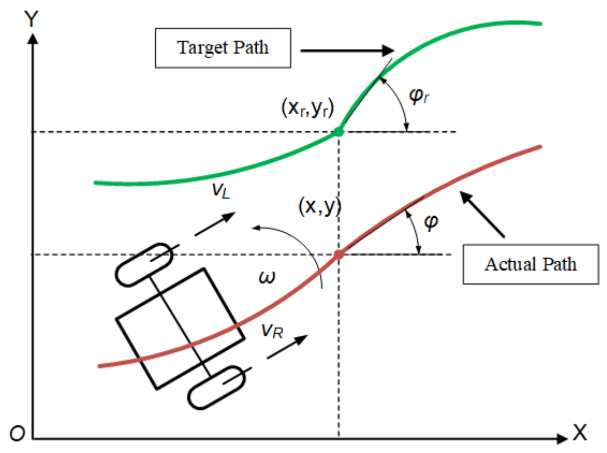

Figure 1. Kinematic model of two-wheels mobile robot.

2.2. Nonlinear Model Predictive Control

The NMPC can be converted into a closed-loop optimal control problem in a finite time-horizon, which is subject to the input vector and state vector constraints. The NMPC formulation can be presented as

$$X(j+1) = \psi_1(X(j)) + \psi_2(X(j))\,u(j) \tag{8}$$

the input and state vector constraints are defined as:

$$X(j) \in R^X, j = 1, 2, \ldots, N$$
$$u(j) \in R^U, j = 1, 2, \ldots N_u$$

where X and u denote state vector and input vector, respectively. In a nonlinear affine system (8), $\psi_1(.)$ and $\psi_2(.)$ denote the nonlinear continuous functions. The initial conditions satisfy $\psi_1(0) = 0$, and $N \geq 1$ and $1 \leq N \leq N_u$ that denotes the prediction horizon.

In each sampling period, the optimal input vector can be obtained by a given cost function online optimization, thus the NMPC scheme is formulated by iterative solution of optimal control problems. The cost function can be described as the following form function:

$$S(X, u) = \sum_{i=j}^{j+N-1} L_1\left(X(i), u(i)\right) + L_2\left(X(j+N)\right) \tag{9}$$

where $L_1(X, u)$ denotes the immediate cost and the satisfying condition is $L_1(0, 0) = 0$. The cost function $S(X, u)$ can be defined as a quadratic form as follows:

$$S(j) = \sum_{i=1}^{N} \|X\left(j+i|j\right)\|_Q^2 + \sum_{i=0}^{N_u-1} \|\Delta u\left(j+i|j\right)\|_R^2 \tag{10}$$

where $X(j+i|j)$ is the predicted future horizon state; $\Delta u(j+i|j) = u(j+i|j) - u(j-1+i|j)$ which is the increment of the input vector; the parameters R and Q represent the constant design weight matrix; the symbol $\|\cdot\|$ represents the Euclidean norm of the corresponding vector. The Equation (8) can be reformulated as:

$$X_e\left(j+1\right) = \psi_1\left(X_e(j)\right) + \psi_2\left(X_e(j)\right) u_e(j) \tag{11}$$

$$\textit{subject to} \qquad u^- \leq u_e(j) \leq u^+ \tag{12}$$
$$\Delta u^- \leq \Delta u(j) \leq \Delta u^+ \tag{13}$$
$$X^- \leq X_e(j) \leq X^+ \tag{14}$$

$$\psi_1\left(X_e\right) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + T \begin{bmatrix} \omega_r x_2 \\ -\omega_r x_1 + v_r x_3 \\ 0 \end{bmatrix}$$

$$\psi_2\left(X_e\right) = T \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

where $X_e = [x_1, x_2, x_3]^T = [x_e, y_e, \theta_e]^T$ is the state vector, and T is the sampling period; $u_e = [u_1, u_2]^T$, $u_1 = v_r cos(\theta_e) - v$, $u_2 = \omega_r - \omega$ is the input vector; (u^-, u^+) are the upper and lower limits of input variable, and $(\Delta u^-, \Delta u^+)$, (x^-, x^+) are also the upper and lower limits of the input increment variable and state variable, respectively.

To construct the QP problem for online optimization, we introduce the vectors as:

$$\bar{X} = [X_e(j+1|j), ..., X_e(j+N|j)]^T \tag{15}$$
$$\bar{u}(j) = [u_e(j|j), ..., u_e(j+N_u-1|j)]^T \tag{16}$$
$$\Delta\bar{u}(j) = [\Delta u_e(j|j), ..., \Delta u_e(j+N_u-1|j)]^T \tag{17}$$

According to (11), and (15)–(17), we can get the predicted output,

$$\bar{X}(j) = H\Delta\bar{u}(j) + \hat{\nu}_1 + \hat{\nu}_2 \tag{18}$$

$$H = \begin{bmatrix} \nu_2\left(X_e(j|j-1)\right) & \cdots & 0 \\ \nu_2\left(X_e(j+1|j-1)\right) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \nu_2\left(X_e(j+N-1|j-1)\right) & \cdots & \nu_2\left(X_e(j+N-1|j-1)\right) \end{bmatrix}$$

$$\hat{\nu}_1 = \begin{bmatrix} \nu_1\left(X_e(j|j-1)\right) \\ \nu_1\left(X_e(j+1|j-1)\right) \\ \vdots \\ \nu_1\left(X_e(j+N-1|j-1)\right) \end{bmatrix}$$

$$\hat{\nu}_2 = \begin{bmatrix} \nu_2\left(X_e(j|j-1)u(k-1)\right) \\ \nu_2\left(X_e(j+1|j-1)u(j-1)\right) \\ \vdots \\ \nu_2\left(X_e(j+N-1|j-1)u(j-1)\right) \end{bmatrix}$$

where $H \in R^{3N \times 2N_u}$, $\hat{\nu}_1 \in R^{3N}$, and $\hat{\nu}_2 \in R^{3N}$. Then, the optimization problem in (10) can be rewritten,

$$\textit{minimum} \quad \|H\Delta\bar{u}(j) + \hat{\nu}_1 + \hat{\nu}_2\|_Q^2 + \|\Delta u(j)\|_R^2 \tag{19}$$
$$\textit{subject to} \quad \Delta\bar{u}^- \leq \Delta\bar{u}(j+1) \leq \Delta\bar{u}^+ \tag{20}$$
$$\bar{u}^- \leq \bar{u}(j-1) \leq \bar{u}^+ \tag{21}$$
$$\bar{u}^- \leq \bar{u}(j-1) + \hat{I}\Delta\bar{u}(j) \leq \bar{u}^+ \tag{22}$$
$$X^- \leq \hat{\nu}_1 + \hat{\nu}_2 + H\Delta\bar{u}(j) \leq X^+ \tag{23}$$

where $\hat{I} = \begin{bmatrix} I & 0 & 0 & 0 \\ I & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & \cdots & I \end{bmatrix} \in R^{2N_u \times 2N_u}$

The optimization problem in (19)–(23) can be reformulated as a QP problem:

$$\textit{minimum} \quad \tfrac{1}{2}\Delta\bar{u}^T M \Delta\bar{u} + c^T\Delta\bar{u} \tag{24}$$
$$\textit{subject to} \quad E\Delta\bar{u} \leq r \tag{25}$$
$$\Delta\bar{u}^- \leq \Delta\bar{u} \leq \Delta\bar{u}^+ \tag{26}$$

and the parameter details are given,

$$M = 2(H^T\eta_1 H + \eta_2) \in R^{2N_u \times 2N_u}$$

$$c = -2H^T \eta_1 (\hat{v_2} + \hat{v_1}) \in R^{2N_u}$$

$$E = \begin{bmatrix} -\hat{I} \\ \hat{I} \\ -H \\ H \end{bmatrix} \in R^{(4N_u+6N)\times 2N_u},\ r = \begin{bmatrix} -u^- + u(j-1) \\ u^+ - u(j-1) \\ -X^- + \hat{v_1} + \hat{v_2} \\ X^+ - \hat{v_1} - \hat{v_2} \end{bmatrix} \in R^{4N_u+6N}$$

3. Varying-Parameter Convergent Differential Neural Network (VPCDNN)

Firstly, we can rewrite the QP problem as follows:

$$\textit{minimum} \quad \tfrac{1}{2} d^T M d + c^T d \tag{27}$$
$$\textit{subject to} \quad Ed \le r \tag{28}$$
$$\zeta^- \le d \le \zeta^+ \tag{29}$$

where $d = \Delta\bar{u}$, $\zeta^- = \Delta\bar{u}^-, \zeta^+ = \Delta\bar{u}^+$, and $d \in [\zeta^-, \zeta^+]$.

In order to find the optimal solution of the time-varying convex quadratic programming problem, we design the Lagrange function for Equations (27)–(29) as follows:

$$L(d, y) = \frac{d^T M d}{2} + c^T d + y^T (Ed - r) \tag{30}$$

where variable y is the Lagrange-multiplier. From the Lagrangian theorem in Reference [29], if the derivatives of $\frac{\partial L}{\partial d}, \frac{\partial L}{\partial y}$ are continuous, we give the following algebraic equations of the Lagrange necessary condition by setting the partial derivatives of $L(d, y)$ to zero,

$$\begin{cases} \frac{\partial L}{\partial d} = Md + c + E^T y = 0 \\ \frac{\partial L}{\partial y} = Ed - r = 0 \end{cases} \tag{31}$$

We can rewrite the equation in (31) as the matrix form,

$$Wp = l \tag{32}$$

$$W = \begin{bmatrix} M & E^T \\ E & 0 \end{bmatrix} \in R^{(6N_u+6N)\times(6N_u+6N)}$$

$$p = \begin{bmatrix} d \\ y \end{bmatrix} \in R^{6N_u+6N}, l = \begin{bmatrix} -c \\ r \end{bmatrix} \in R^{6N_u+6N}$$

where W is a square matrix and invertible; p denotes the vector that needs to be solved by physical limits. It should be noted that the vectors c and r are time-varying parameters. Therefore, the time-varying solver is considered to solve the time-varying QP problem in (27)–(29).

Actually, the optimal solution is under the condition that the equation in (32) is equivalent to zero. So, the error expression is defined as

$$e(t) = Wp(t) - l(t) \tag{33}$$

where the error $e(t) \in R^{6Nu+6N}$ and we hope the variable e is close to zero. Therefore, the varying-parameter convergent differential neural network is proposed for the problem in (33),

$$\dot{e}(t) = -\lambda \exp(\kappa t) P_{\Omega}(e(t)) \tag{34}$$

$$P_{\Omega}(e(t)) = \begin{cases} e_i^-, \text{if } e_i < e_i^- \\ e_i, \text{if } e_i^- \le e_i \le e_i^+, \forall i \in \{1, \cdots, 6N_u + 6N\} \\ e_i^+, \text{if } e_i > e_i^+ \end{cases}$$

where the parameters λ and κ are positive constants, which are used to scale the convergence rate; the symbol $P_{\Omega}(\cdot)$ is the active function. Then, substituting (33) into (34), the differential equation of the solver for the QP problem can be concluded as

$$W\dot{p}(t) = -\dot{W}p(t) - \lambda \exp(\kappa t) P_{\Omega}(Wp(t) - l(t)) + \dot{l}(t) \tag{35}$$

The block diagram of the VPCDNN model (35) is shown in Figure 2. Finally, for the online solving process, the neural network consists of N_p neurons and the neural network is constructed as

$$\dot{p}_i = -\sum_{j=1}^{N_p} \dot{w}_{ij} p_i + \sum_{j=1}^{N_p} (\rho_{ij} - w_{ij}) \dot{p}_i - \lambda \exp(\kappa t) P_{\Omega} \left\{ \sum_{j=1}^{N_p} (w_{ij} p_i - l_i) \right\} + \dot{l}_i \tag{36}$$

where p_i, $\dot{p}_i$, l_i, $\dot{l}_i$ are the i-th elements of variable p, $\dot{p}$, l_i, $\dot{l}_i$, respectively; w_{ij},$\dot{w}_{ij}$, η_{ij},$\dot{\eta}_{ij}$ are the i-th row and j-th column elements of variable W, I, respectively. Figure 3 exhibits the detailed structure of the VPCDNN model.

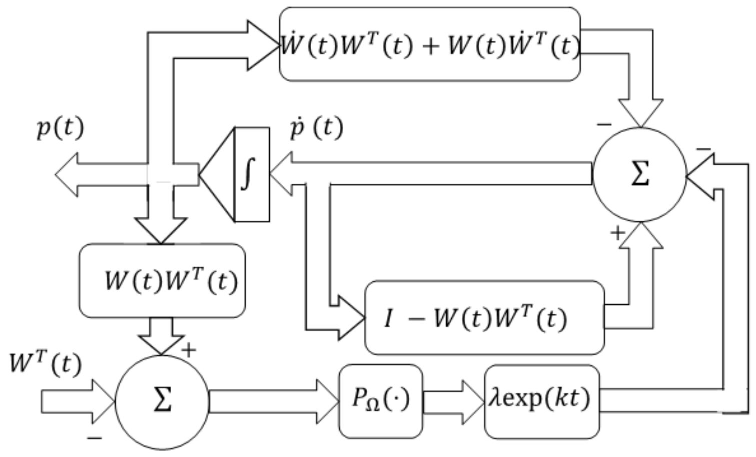

Figure 2. The block diagram of VPCDNN model.

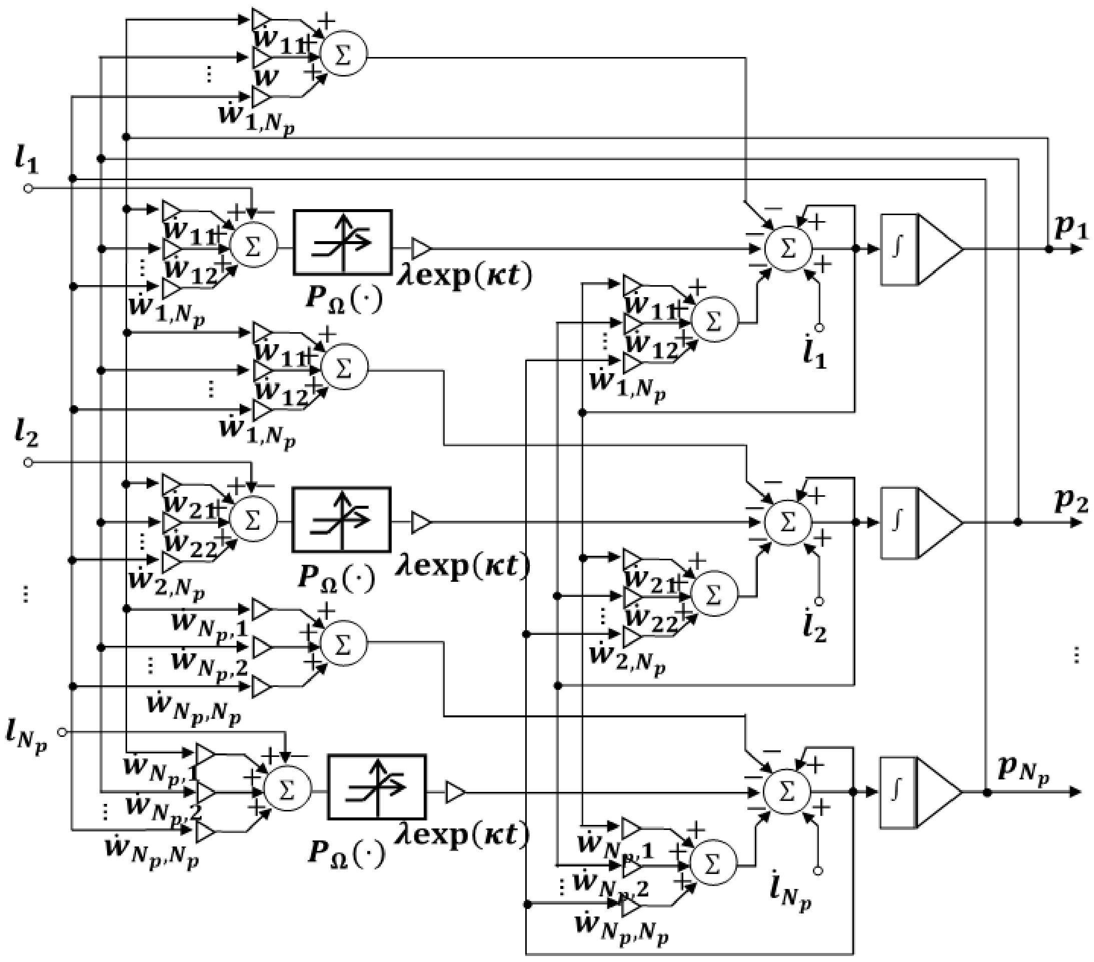

Figure 3. Structure of VPCDNN model.

4. Convergence and Robustness Analysis of VPCDNN

4.1. Convergence Analysis

It should be noted that the VPCDNN algorithm is a special type of Hopfield-neural network because the network fits the Hopfield-neural network structure with reference to the differential equation theory.

If there exists the optimal solution d^* for (27)–(29), the vector p converges to equilibrium point p^* from all initial states $p(0)$, which means the VPCDNN can achieve the convergence and the stable point p^* of VPCDNN is the optimal solution of the QP problem. Firstly, we design the Lyapunov function of error variable $e(t)$ and then find the minimum point of the Lyapunov function. The detailed proof of convergence is given as follows.

Theorem 1. *For the time-varying QP problem in (27)–(29), assume that there is the optimal solution d^* for the QP problem when the monotone nondecreasing activation function of P_Ω acts on the error variable. From any initial state $p(0) \in R^{6N_u+6N}$, the state vector $p(t) = [d^T(t), y^T(t)]^T$ of VPCDNN globally converges to equilibrium point $p^*(t) = \left[d^{*T}(t), y^{*T}(t)\right]^T$, where the first n-elements of p^* (x^*) are the optimal solution of (27)–(29).*

Proof. Considering the candidate Lyapunov function as

$$F(t) = \frac{1}{2}e^T(t)e(t),\ \forall e(t) \neq 0 \tag{37}$$

It is obvious that the Lyapunov function in (37) is a positive function. The time derivative of $F(t)$ can be obtained as

$$\dot{F}(t) = \frac{dF(t)}{dt} = e^T(t)\dot{e}(t) \tag{38}$$

Then, substituting (34) into (38), we obtain

$$\begin{aligned}\dot{F}(t) &= -\lambda \exp(\kappa t)e^T(t)P_\Omega\left\{e(t)\right\} \\ &= -\lambda \exp(\kappa t)\sum_{i=1}^{N_p} e_i(t)P_\Omega\left(e_i(t)\right)\end{aligned} \tag{39}$$

where $e_i(t)$ represents the ith element of variable $e(t)$; $P_\Omega\left\{e(t)\right\}$ is the ith projection element of variable $P_\Omega\left\{e(t)\right\}$. Since $P_\Omega\left\{e(t)\right\}$ is the monotone nondecreasing projection function, it is easy to obtain

$$e_i(t)P_\Omega\left(e_i(t)\right) = \begin{cases} > 0, \text{if } e_i(t) \neq 0 \\ = 0, \text{if } e_i(t) = 0 \end{cases} \tag{40}$$

Therefore, we can obtain the following,

$$\dot{F}(t) = \begin{cases} > 0, \text{if } e_i(t) \neq 0 \\ = 0, \text{if } e_i(t) = 0 \end{cases} \tag{41}$$

According to the (41), we know that if and only if $e_i(t) = 0$, $\dot{F}(t)$ is equal to zero. Therefore, $p(t) - p^*(t)$ globally converges to 0 based on Lyapunov theory [30], which also means $d(t) - d^*(t)$ globally converges to zero. So, the proof of VPCDNN is finished. □

4.2. Robustness Analysis

Considering the disturbance model in (35) as follows,

$$W\dot{p}(t) = -(\dot{W} + \Delta B(t))p(t) - \lambda \exp(\kappa t)P(Wp(t) - l(t)) + \dot{l}(t) + \Delta\xi(t) \tag{42}$$

where $\Delta B \in R^{(6N_u+6N)\times(6N_u+6N)}$ is the disturbing term of W; $\Delta\xi \in R^{n+m}$ is the error from the VPCDNN model.

Theorem 2. *If $\|\Delta B(t)\| \leq \mu_B$, $\|\Delta\xi(t)\| \leq \mu_\xi$, $\left\|W^{-1}\right\| \leq \mu_W$, $\|l(t)\| \leq \mu_l$, and μ_B, where μ_ξ, μ_W, μ_l are all positive parameters, the error $e(t)$ will converge to* 0 *under the condition of $\lambda a \exp(\kappa t) - \mu_B\mu_A > 0$.*

Proof. Substituting the (33), (34) into (42), we can conclude

$$\dot{e}(t) = -\lambda \exp(\kappa t)P_\Omega\left(e(t)\right) - \Delta B(t)W^{-1}e(t) + \Delta\xi(t) - \Delta B(t)W^{-1}l(t) \tag{43}$$

We choose a Lyapunov function as

$$V(t) = \frac{1}{2}e^T(t)e(t) = \frac{1}{2}\sum_{j=1}^{6N_u+6N} e_i^2(t) \tag{44}$$

where $V(t)$ is the non-negative variable.

Then, we can obtain the time derivative of $V(t)$,

$$\begin{aligned} &V(t) = e^T(t)\dot{e}(t) \\ &= e^T(t)(-\lambda \exp(\kappa t)P_\Omega(e(t)) - \Delta B(t)W^{-1}e(t) + \Delta\xi(t) - \Delta B(t)W^{-1}l(t)) \\ &= -\lambda \exp(\kappa t)e^T(t)P_\Omega(e(t)) + e^T(t)\psi(t)e(t) + e^T(t)\Delta\xi(t) + e^T(t)\left(-\Delta B(t)W^{-1}l(t)\right) \\ &= -\lambda \exp(\kappa t)e^T(t)P_\Omega(e(t)) + e^T(t)\tfrac{\psi(t)+\psi^T(t)}{2}e(t) + e^T(t)\Delta\xi(t) + e^T(t)\left(-\Delta B(t)W^{-1}l(t)\right) \end{aligned} \tag{45}$$

where $\psi(t) = -\Delta B(t)W^{-1}$. Moreover,

$$\begin{aligned} &e^T(t)\tfrac{\psi(t)+\psi^T(t)}{2}e(t) \le e^T(t)e(t)\left|\lambda_{\max}\left(\tfrac{\psi(t)+\psi^T(t)}{2}\right)\right| \\ &\le e^T(t)e(t)\left\|\Delta B(t)W^{-1}(t)\right\| \le e^T(t)e(t)\mu_B\mu_W \end{aligned} \tag{46}$$

$$e^T(t)\Delta\xi(t) \le \sum_{i=1}^{6N_u+6N} |e_i|\mu_\xi \tag{47}$$

$$e^T(t)\left(-\Delta B(t)W^{-1}l(t)\right) \le \sum_{i=1}^{6N_u+6N} |e_i| \cdot \left\|-\Delta B(t)W^{-1}l(t)\right\| \le \sum_{i=1}^{6N_u+6N} |e_i|\mu_B\mu_\xi\mu_l \tag{48}$$

Therefore, substituting (46)–(47) into (45),

$$\begin{aligned} &\dot{V}(t) \le -\lambda \exp(\kappa t)e^T(t)P_\Omega\left(e(t)\right) + e^T(t)e(t)\mu_B\mu_W + \sum_{i=1}^{6N_u+6N}|e_i|\mu_\xi + \sum_{i=1}^{6N_u+6N}|e_i|\mu_B\mu_\xi\mu_l \\ &= -\sum_{i=1}^{6N_u+6N}|e_i|\left(\lambda \exp(\kappa t)P_\Omega(|e_i|) - \mu_B\mu_W|e_i| - \mu_\xi - \mu_B\mu_\xi\mu_l\right) \end{aligned} \tag{49}$$

We define the $\Theta_1 = \lambda \exp(\kappa t)P_\Omega(|e_i|) - \mu_B\mu_W|e_i| - \mu_\xi - \mu_B\mu_\xi\mu_l$. We know variable Θ_1 may be positive or negative.

1. if $\Theta_1 \ge 0$, so $\dot{V} \le 0$. It is obvious that the error variable $e(t)$ converges to zero from the Lyapunov theorem and the state variable p converges to the optimal solution p^*.
2. if $\Theta_1 < 0$, then $\dot{V} < \delta(\delta > 0)$. Therefore, $\dot{V}$ may be positive or negative.
   (a) if $\dot{V} \le 0$, we know the error variable $e(t)$ converges to zero, also the state variable p will converge to optimal solution p^*.
   (b) if $\dot{V} > 0(0 < \dot{V} < \delta)$, and consider the linear activation function $\Theta_1(|e(t)|) = a|e(t)|\,(a \ge 1)$, and $\lambda a \exp(\kappa t) - \mu_B\mu_W > 0$, so we can obtain,

$$\begin{aligned} &\dot{V} \le -\sum_{i=1}^{6N_u+6N}|e_i|\left(\lambda \exp(\kappa t)a|e_i| - \mu_B\mu_W|e_i| - \mu_\xi - \mu_B\mu_W\mu_l)\right) \\ &= -\left(\lambda a \exp(\kappa t) - \mu_B\mu_W\right)\sum_{i=1}^{n+m}|e_i|\left(|e_i| - \tfrac{\mu_\xi+\mu_B\mu_W\mu_l}{\lambda a \exp(\kappa t)-\mu_B\mu_W}\right) \end{aligned} \tag{50}$$

It is easy to obtain $\lambda > \frac{\mu_B \mu_W}{a}$. According to (50), in this case, $0 < \dot{V} < \delta$, so $V(t)$ increase that means $|e_i|$ will increase, and $\dot{V}$ will decrease. Therefore, $\dot{V}$ always exists a moment $\dot{V} \leq 0$, then the control system will stabilize again.

It should be noted that when $\dot{v}(t) = 0$, $|e_i| = |e|^+$, $|e|^+$ is the upper bound. We define $\Theta_2 = \frac{\mu_\xi + \mu_B \mu_W \mu_l}{\lambda a \exp(\kappa t) - \mu_B \mu_W}$. If $\dot{V}(t) = 0$, $\sum\limits_{i=1}^{6N_u+6N} |e_i| \left(|e_i| - \Theta_2\right) = 0$ and the variable $|e_i|$ can be seen as the input, the function $|e_i| \left(|e_i| - \Theta_2\right)$ will obtain the minimum output, if $|e_i| = 0.5\Theta_2$. Moreover, $|e_i| \left(|e_i| - \Theta_2\right) > 0$ when $|e_i| > \Theta_2$

We know $\sum\limits_{i=1}^{6N_u+6N} |e_i| \left(|e_i| - \Theta_2\right) = 0$ and the function mentioned above has a negative minimum output. We assume that $e_j (i = j)$ is the upper bound e^+, so e_j will be achieved if and only if the rest $6N_u + 6N - 1$ terms $|e_i| \left(|e_i| - \Theta_2\right)$ obtain the minimum point. Therefore,

$$\sum_{i=1}^{6N_u+6N} |e_i| \left(|e_i| - \Theta_2\right) = \sum_{i=1, i \neq j}^{6N_u+6N} |e_i| \left(|e_i| - \Theta_2\right) + |e_j| \left(|e_j| - \Theta_2\right) \tag{51}$$

Then, $\Theta_2 = \frac{\mu_\xi + \mu_B \mu_W \mu_l}{\lambda a \exp(\kappa t) - \mu_B \mu_W}$ is substituted in (51) and obtain

$$\sum_{i=1}^{6N_u+6N} |e_i| \left(|e_i| - \Theta_2\right) = |e_j|^2 - |e_j| \left(\frac{\mu_\xi + \mu_B \mu_W \mu_l}{\lambda a \exp(\kappa t) - \mu_B \mu_W}\right) - \frac{6N_u + 6N - 1}{4} \left(\frac{\mu_\xi + \mu_B \mu_W \mu_l}{\lambda a \exp(\kappa t) - \mu_B \mu_W}\right)^2 = 0 \tag{52}$$

According to analysis, we can conclude the upper bound $|e_j|$,

$$|e_j| = \frac{1}{2} \left((1 + \sqrt{6N_u + 6N}) \eta \right)$$

where $\eta = \frac{\mu_\xi + \mu_B \mu_W \mu_l}{\lambda a \exp(\kappa t) - \mu_B \mu_W}$, $|e_{end}|$ $(end = 6N_u + 6N)$ converge to zero. $\square$

5. Simulation

In this section, the proposed VPCDNN algorithm is tested with a two-wheel mobile robot in simulation. Two type trajectories will be tested for tracking: circle-shape trajectory (the linear velocity v and angular velocity w are time constant); '8'-shape trajectory (the linear velocity v and angular velocity w are time varying) and the trajectory function in Cartesian space will be given in Section 5.2. Both of the two trajectories will be tested in mobile robot tracking and the detailed analysis of the simulation results will be given in Sections 5.1 and 5.2.

5.1. Circle-Shape Tracking

In this part, we control the mobile robot to track the circle-shape trajectory, where the reference linear velocity v and angular velocities ω are set as $v = 0.2$ m/s, $w = 0.2$ rad/s, respectively. In order to prove the tracking performance of our method, the initial position of the mobile robot is set as $X_r(0) = [1.0; 0; 0]$ which does not coincide with the reference initial point $X(0) = [1.2; 0; 0]$.

The parameters of nonlinear MPC are listed as: $N = 3$, $N_u = 2$, $\eta_1 = 3I$, $\eta_2 = I$. The neural network parameter λ and κ are set as: $\lambda = 0.1$, $\kappa = 1e-4$. The sampling period of MPC is $\Delta t = 0.1s$.

From Figures 4 and 5, it can be seen that the trajectories of the mobile robot quickly converge to the reference trajectory, regardless of the initial position of the mobile robot. From Figures 6 and 7, we can see the mobile robot's linear and angular velocity are extremely close to the reference values. We can also see that the tracking errors of the mobile robot quickly converge to zero as shown in Figures 8 and 9.

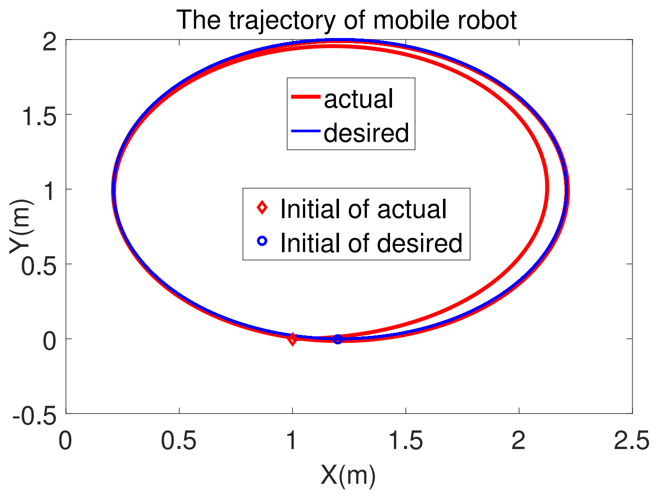

Figure 4. Tracking result of circle-shape trajectory.

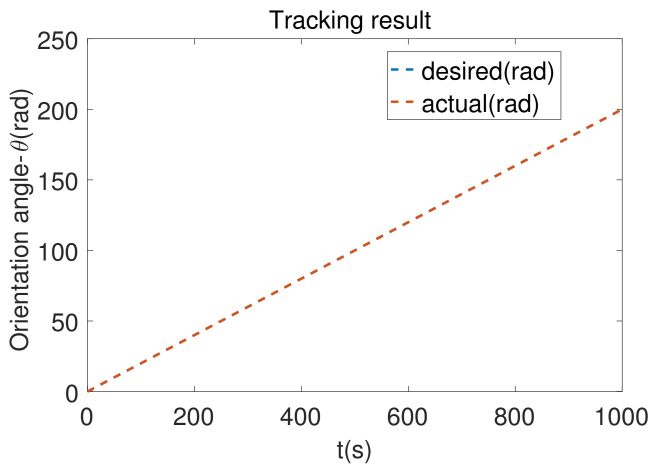

Figure 5. Tracking result of orientation angle.

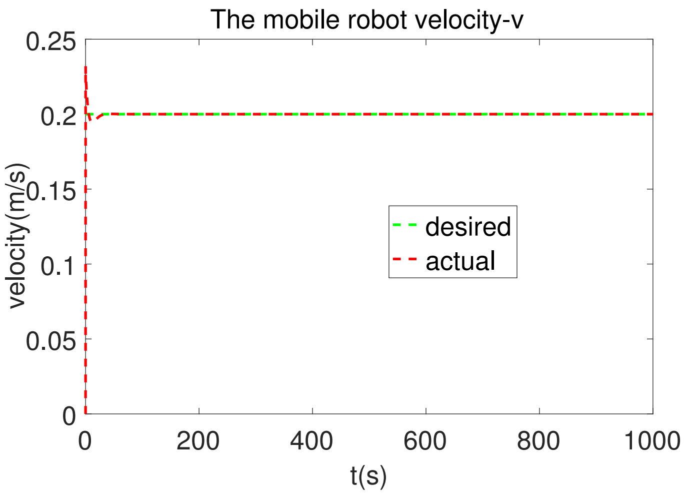

Figure 6. Linear velocity of mobile robot.

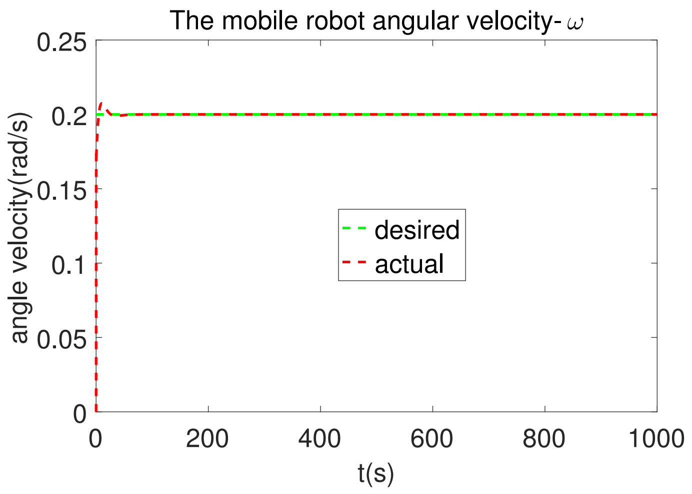

Figure 7. Angular velocity of mobile robot.

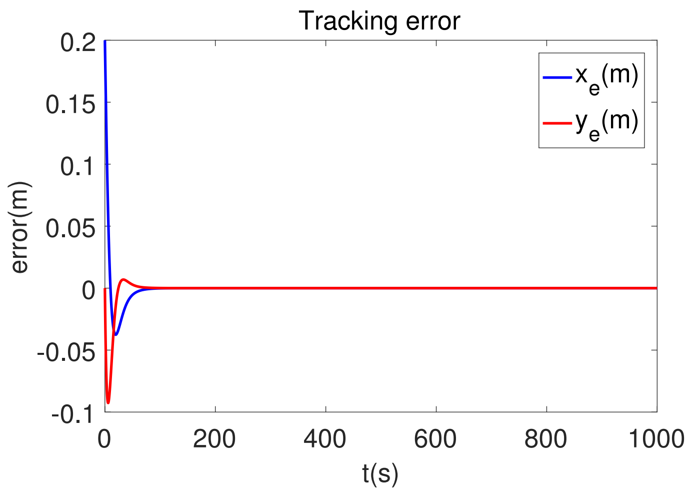

Figure 8. Tracking error in Cartesian space.

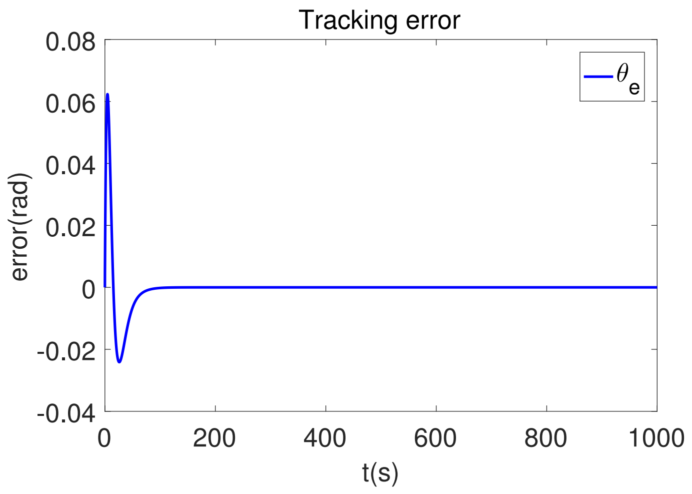

Figure 9. Tracking error of orientation angle.

5.2. '8'-Shape Tracking

The mobile robot's task in this scenario is to track the '8'-shape trajectory for the periodic movement with changing linear and angular velocity, starting from initialized positions and orientations. The reference trajectory of '8'-shape is given as

$$x = sin(0.1t)$$
$$y = 2sin(0.05t)$$

The parameters of nonlinear MPC are listed as: $N = 3$, $N_u = 2$, $\eta_1 = 3I$, $\eta_2 = I$. The convergence parameter λ and κ are set as: $\lambda = 0.1$, $\kappa = 1e - 4$. The sampling period of MPC is set $\Delta t = 0.1$ s.

The initial position of the reference trajectory is $X_r(0) = [0; 0; 0.7854]$, and the initial position of the mobile robot is $X(0) = [0.3; 0; 0.7]$. From Figures 10 and 11, we can see that the trajectories of the mobile robot quickly converge to the reference trajectory even changing the linear and angular velocity of the robot periodically. From Figures 12 and 13, we can see that the mobile robot's linear and angular velocity follow the reference velocity quite well. Figures 14 and 15 show that the tracking errors of the mobile robot quickly converge to zero.

In both scenarios, the mobile robot is able to track the reference trajectory successfully. The convergence rate is very fast which is very important for running in real time. The mobile robot solves the trajectory tracking task with the optimal solution from different initialized positions and orientations. The tracking errors also converge very fast to zero. These results show the robustness and effectiveness of the proposed method. Moreover, the overall success rate, fast convergence, and the performance of the proposed VPCDNN based MPC method indicates its strong ability in real time.

In order to verify the effectiveness of VPCDNN, we have added contrast tests using LNN methods which include the time- invariant method called the recurrent neural network (gradient-descent based) in Reference [31]. The comparison results are shown in Figures 16–18. We can conclude that both VPCDNN and RNN can track the given trajectory, but VPCDNN has the faster convergence speed which is better for the time-varying optimization problem.

In addition, in this paper, we connect CarSim software to a Matlab/Simulink model and test the VPCDNN-MPC method to track the sinusoidal trajectory which is shown in Figure 19. The requirements of simulator platform were Matlab2018b/Simulink, Carsim2016, Windows10.

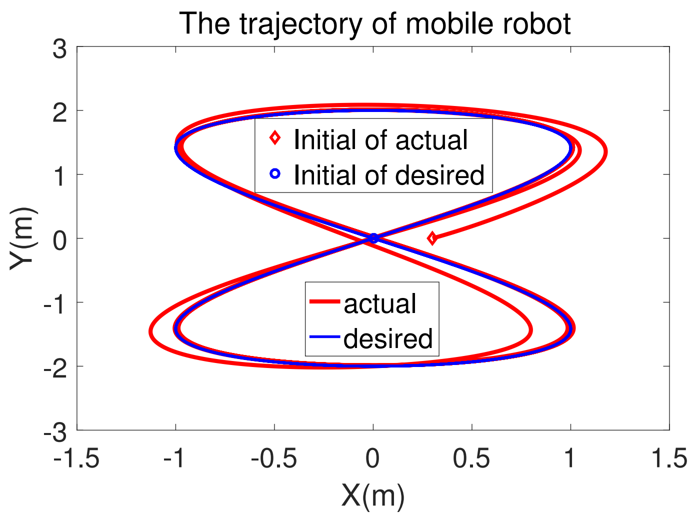

Figure 10. Tracking result of '8'-shape trajectory.

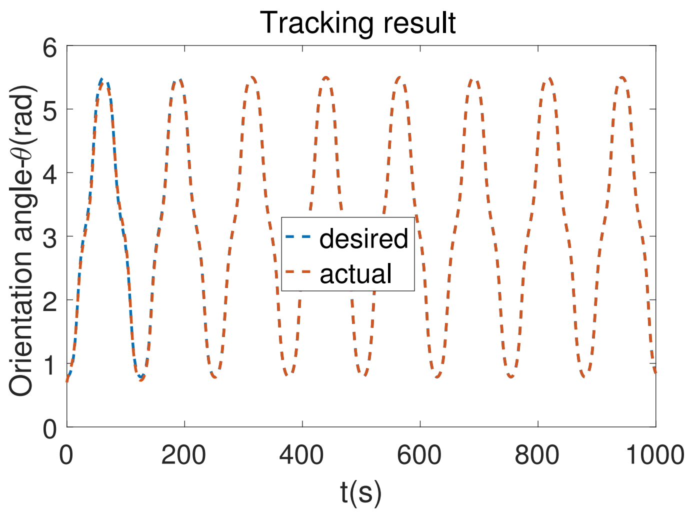

Figure 11. Tracking result of orientation angle.

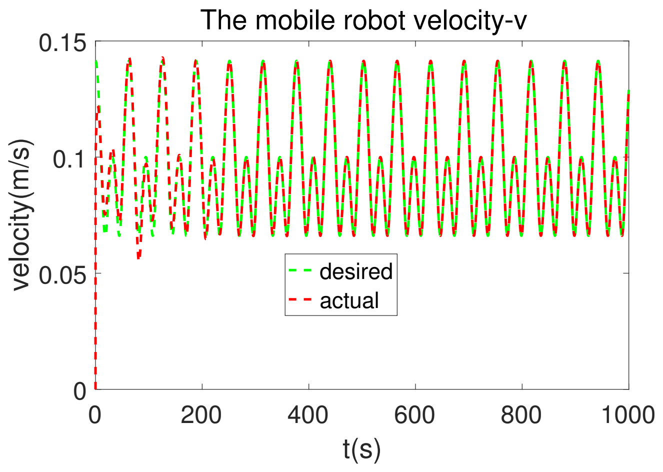

Figure 12. Linear velocity of mobile robot.

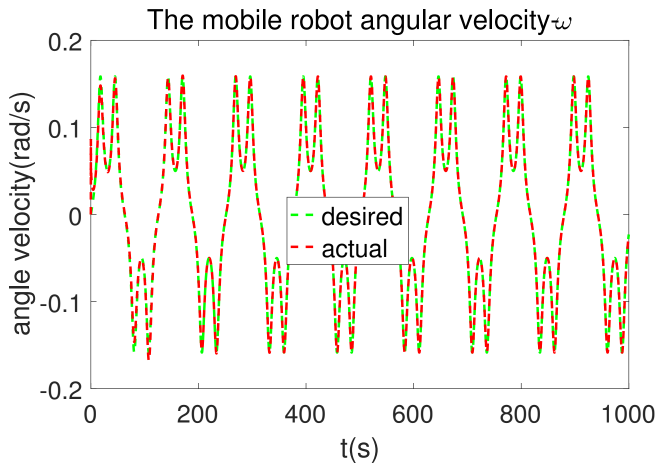

Figure 13. Angular velocity of mobile robot.

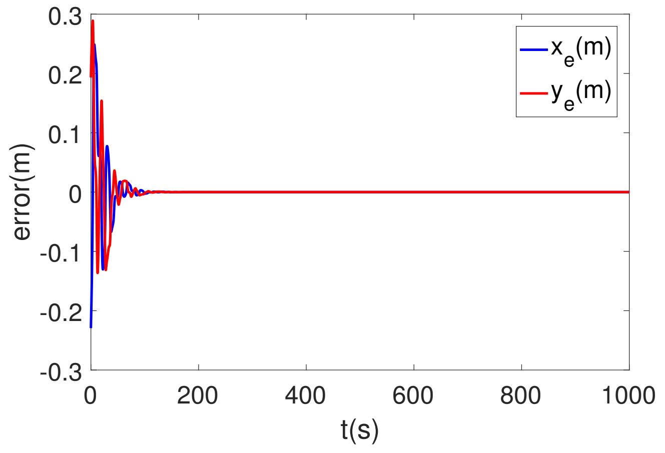

Figure 14. Tracking error in Cartesian space.

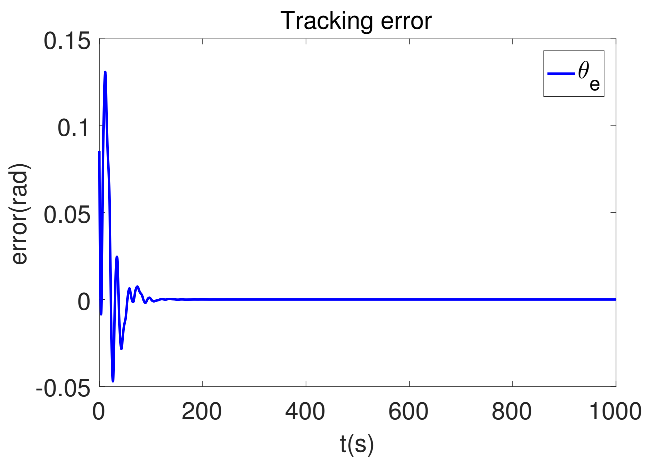

Figure 15. Tracking error of orientation angle.

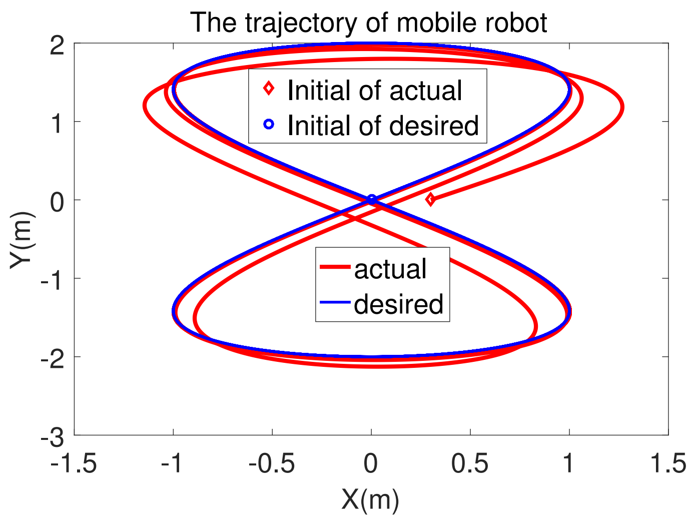

Figure 16. Comparison results of '8'-shape using LNN.

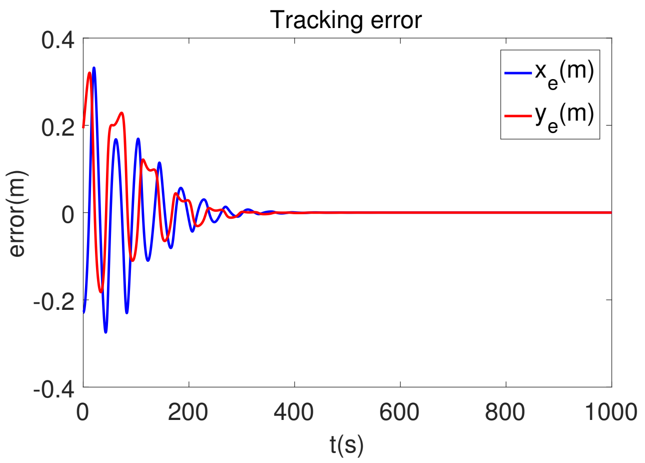

Figure 17. Comparison results of X_e, Y_e error using LNN.

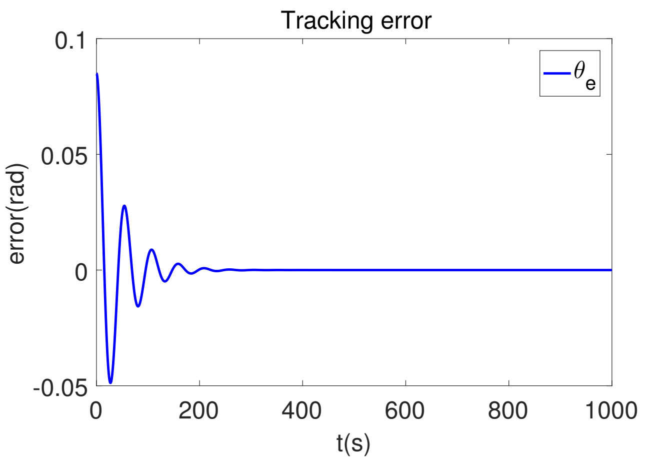

Figure 18. Comparison results of θ_e error using LNN.

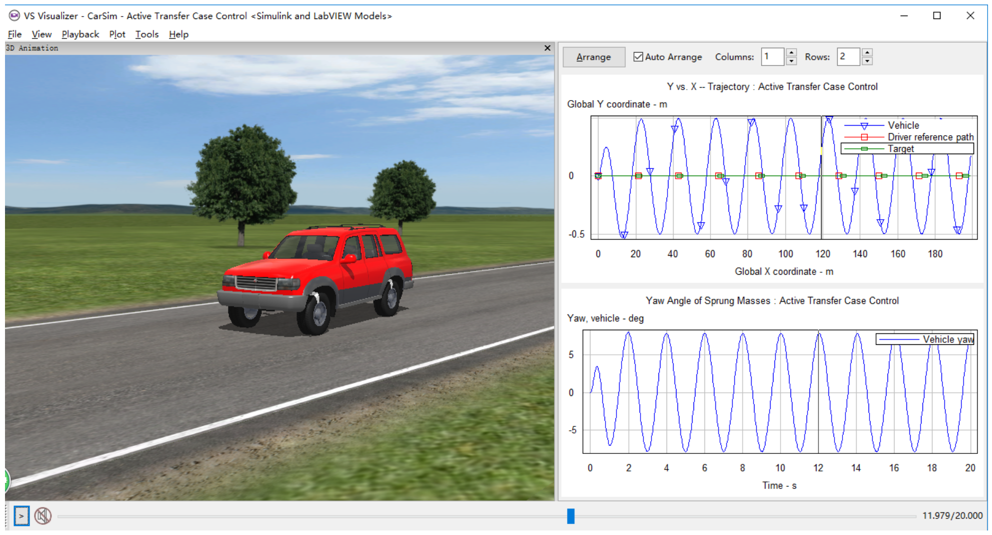
VS Visualizer - CarSim - Active Transfer Case Control <Simulink and LabVIEW Models>
File View Playback Plot Tools Help
3D Animation
Arrange
Auto Arrange
Columns: 1
Rows: 2
Y vs. X -- Trajectory : Active Transfer Case Control
Global Y coordinate - m
Vehicle
Driver reference path
Target
Global X coordinate - m
Yaw Angle of Sprung Masses : Active Transfer Case Control
Yaw, vehicle - deg
Vehicle yaw
Time - s
11.979/20.000

Figure 19. Tracking the sinusoidal trajectory using vehicle in Carsim simulation.

6. Conclusions

In this paper, the MPC scheme is presented to the control problem over finite time-horizon with the input and state vector of a mobile robot. In the reformation of the nonlinear affine system of a mobile robot, the nonlinear non-convex optimization problem is converted to the nonlinear convex optimization problem. Therefore, the new network of VPCDNN is also presented to solve the online optimization QP problem of NMPC. In order to test the proposed method, the mobile robot tracks the circle-shape and '8'-shape trajectories in simulation. The results show that the proposed method successfully tracked the desired trajectories. The method converges quickly to the solution with the VPCDNN. The tracking errors of the VPCDNN based NMPC are maintained at a very low level. All these characteristics enable our proposed method to be a powerful and efficient solution to the control problem of mobile robots.

Author Contributions: Conceptualization, Y.H. and G.C.; methodology, Y.H. and H.S.; software Y.H. and S.M.; validation, L.Z. and H.S.; formal analysis, S.M. and H.S.; investigation, data curation, L.Z.; writing-original draft preparation,Y.H.; writing-reviewand editing, G.C. and H.S.; supervision, A.K. and G.C.; project administration, A.K.

Funding: This work was supported by the German Research Foundation (DFG) and the Technical University of Munich (TUM) in the framework of the Open Access Publishing Program. This research also received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 785907 (Human Brain Project SGA2).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

NMPC	nonlinear model predictive control
VPCDNN	varying-parameter convergent differential neural network
QP	quadratic programming
SMC	sliding mode control
GPNN	general projection neural network
PDNN	primal dual neural network

References

1. Brockett, R.W. Asymptotic stability andfeedback stabilization. *Differ. Geom. Control Theory* **1983**, *27*, 181–208.
2. Al-Araji, A.S. Development of kinematic path-tracking controller design for real mobile robot via back-stepping slice genetic robust algorithm technique. *Arab. J. Sci. Eng.* **2014**, *39*, 8825–8835. [CrossRef]
3. Fukushima, H.; Muro, K.; Matsuno, F. Sliding-mode control for transformation to an inverted pendulum mode of a mobile robot with wheel-arms. *IEEE Trans. Ind. Electron.* **2014** *62*, 4257–4266. [CrossRef]
4. Ostafew, C.J.; Schoellig, A.P.; Barfoot, T.D. Visual teach and repeat, repeat, repeat: Iterative learning control to improve mobile robot path tracking in challenging outdoor environments. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 176–181.
5. Li, T.H.; Chang, S.J.; Tong, W. Fuzzy target tracking control of autonomous mobile robots by using infrared sensors. *IEEE Trans. Fuzzy Syst.* **2004**, *12*, 491–501. [CrossRef]
6. Khalaji, A.K.; Moosavian, S.A.A. Robust adaptive controller for a tractor–trailer mobile robot. *IEEE/ASME Trans. Mechatron.* **2013**, *19*, 943–953. [CrossRef]
7. Park, B.S.; Yoo, S.J.; Park, J.B.; Choi, Y.H. A simple adaptive control approach for trajectory tracking of electrically driven nonholonomic mobile robots. *IEEE Trans. Control Syst. Technol.* **2010**, *18*, 1199–1206. [CrossRef]
8. Armesto, L.; Girbés, V.; Sala, A.; Zima, M.; Šmídl, V. Duality-based nonlinear quadratic control: Application to mobile robot trajectory-following. *IEEE Trans. Control Syst. Technol.* **2015**, *23*, 1494–1504. [CrossRef]

9. Hendzel, Z.; Penar, P. Optimal Control of a Wheeled Robot. In *Conference on Automation*; Springer: Cham, Switzerland, 2019; pp. 473–481.
10. Patrinos, P.; Bemporad, A. An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *IEEE Trans. Autom. Control*. **2013**, *59*, 18–33. [CrossRef]
11. Frison, G.; Sørensen, H.H.B.S.; Dammann, B.; Jørgensen, J.B.J. High-performance small-scale solvers for linear model predictive control. In Proceedings of the 2014 European Control Conference (ECC), Strasbourg, France, 24–27 June 2014; pp. 128–133.
12. Nascimento, T.P.; Dórea, C.E.T.; Goncalves, L.M.G. Nonholonomic mobile robots' trajectory tracking model predictive control: A survey. *Robotica* **2018**, *36*, 676–696. [CrossRef]
13. Nascimento, T.P.; Moreira, A.P.; Conceição, A.G.S. Multi-robot nonlinear model predictive formation control: Moving target and target absence. *Robot. Auton. Syst.* **2013**, *61*, 1502–1515. [CrossRef]
14. Vajedi, M.; Azad, N.L. Ecological adaptive cruise controller for plug-in hybrid electric vehicles using nonlinear model predictive control. *IEEE Trans. Intell. Transp. Syst.* **2015**, *17*, 113–122. [CrossRef]
15. Lima, P.U.; Ahmad, A.; Dias, A.; Conceição, A.G.; Moreira, A.P.; Silva, E.; Nascimento, T.P. Formation control driven by cooperative object tracking. *Robot. Auton. Syst.* **2015**, *63*, 68–79. [CrossRef]
16. Nascimento, T.P.; Dórea, C.E.T.; Goncalves, L.M.G. Nonlinear model predictive control for trajectory tracking of nonholonomic mobile robots: A modified approach. *Int. J. Adv. Robot. Syst.* **2018**, *15*, 1729881418760461. [CrossRef]
17. Mesbah, A. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Syst. Mag.* **2016**, *36*, 30–44.
18. Nascimento, T.P.; Basso, G.F.; Dorea, C.; Goncalves, L.M.G. Perception-driven Motion Control Based on Stochastic Nonlinear Model Predictive Controllers. *IEEE/ASME Trans. Mechatron.* **2019**, *1*, 1–11. [CrossRef]
19. Rubagotti, M.; Patrinos, P.; Bemporad, A. Stabilizing linear model predictive control under inexact numerical optimization. *IEEE Trans. Autom. Control* **2014**, *59*, 1660–1666. [CrossRef]
20. Park, H.; Sun, J.; Pekarek, S.; Stone, P.; Opila, D.; Meyer, R.; DeCarlo, R. Real-time model predictive control for shipboard power management using the IPA-SQP approach. *IEEE Trans. Control Syst. Technol.* **2015**, *23*, 2129–2143. [CrossRef]
21. Sekiguchi, S.; Yorozu, A.; Kuno, K.; Okada, M.; Watanabe, Y.; Takahashi, M. Nonlinear model predictive control for two-wheeled service robots. In *International Conference on Intelligent Autonomous Systems*; Springer: Cham, Switzerland, 2018; pp. 452–463.
22. Pan, Y.; Wang, J. Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. *IEEE Trans. Ind. Electron.* **2012**, *59*, 3089–3101. [CrossRef]
23. Li, Z.; Xiao, H.; Yang, C.; Zhao, Y. Model predictive control of nonholonomic chained systems using general projection neural networks optimization. *IEEE Trans. Syst. Man Cybern. Syst.* **2015**, *45*, 1313–1321. [CrossRef]
24. Xiao, H.; Li, Z.; Chen, C.P. Formation control of leader-follower mobile robots' systems using model predictive control based on neural-dynamic optimization. *IEEE Trans. Ind. Electron.* **2016**, *63*, 5752–5762. [CrossRef]
25. Ke, F.; Li, Z.; Yang, C. Robust Tube-Based Predictive Control for Visual Servoing of Constrained Differential-Drive Mobile Robots. *IEEE Trans. Ind. Electron.* **2018**, *65*, 3437–3446. [CrossRef]
26. Li, Z.; Yuan, Y.; Ke, F.; He, W.; Su, C. Robust Vision-Based Tube Model Predictive Control of Multiple Mobile Robots for Leader-Follower Formation. *IEEE Trans. Ind. Electron.* **2019**, in press, doi:10.1109/TIE.2019.2913813. [CrossRef]
27. Zhang, Y.; Wang, J. A dual neural network for convex quadratic programming subject to linear equality and inequality constraints. *Phys. Lett. A* **2002**, *298*, 271–278. [CrossRef]
28. Hu, Y.; Li, Z.; Li, G.; Yuan, P.; Yang, C.; Song, R. Development of sensory-motor fusion-based manipulation and grasping control for a robotic hand-eye system. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *7*, 1169–1180. [CrossRef]
29. Boyd, S.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004.

30. Coddington, E.A.; Levinson, N.; Teichmann, T. Theory of ordinary differential equations. *Phys. Today* **1956**, doi:10.1063/1.3059875. [CrossRef]
31. Xia, Y.; Wang, J.; Fok, L.M. Grasping-force optimization for multifingered robotic hands using a recurrent neural network. *IEEE Trans. Robot. Autom.* **2004**, *20*, 549–554. [CrossRef]